НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: Масштабовані програмні засоби. Підсистема збору та обробки інформації про виняткові ситуації.

Виконав: студент IV курсу, групи КВ-51
(шифр групи)

Місячний Ігор Валерійович
(прізвище, ім’я, по батькові) (підпис)

Керівник доц. каф. СПСКС, к.т.н Петрашенко А.В
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____
_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент_____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студента
Місячний Ігор Валерійович
(прізвище, ім’я, по батькові)

1. Тема проекту

Масштабовані програмні засоби. Підсистема збору та обробки інформації про виняткові ситуації, керівник проекту Петрашенко Андрій Васильович к.т.н. доцент, затверджені наказом по університету від «22» травня 2019 р. №1330-С _____

2. Термін подання студентом проекту «_ » червня 2019 р.

3. Вихідні дані до проекту див. Технічне завдання

4. Зміст пояснювальної записки

- Аналіз існуючих рішень та обґрунтування теми дипломного проекту
- Структура програмних засобів
- Опис розроблених алгоритмів
- Тестування та результати

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Структурна схема проекту
- Алгоритм роботи системи
- Процес розпізнавання обличчя . Схема алгоритму
- Алгоритм роботи системи моніторингу

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М. доцент		

7. Дата видачі завдання «_ » _____2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вивчення літератури за тематикою проекту	17.11.2018	
2	Розроблення та узгодження технічного завдання	28.11.2018	
3	Аналіз існуючих рішень	15.12.2018	
4	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5	Розроблення програмного забезпечення	03.02.2019	
6	Відлагодження програмного продукту	10.02.2019	
7	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
8	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
9	Підготовка графічної частини дипломного проекту	19.05.2019	
10	Оформлення документації дипломного проекту	30.05.2019	

Студент _____ Місячний І.В
(підпис) (ініціали, прізвище)

М

Керівник проекту _____ Петрашенко А.В.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Об’єкт розробки - це підсистема для засобів моніторингу об'єктів охорони. Головним призначенням якої - збір та обробка інформації про виняткові ситуації.

Система дозволяє: реєструвати та авторизовувати користувачів у додатку; спостерігати за об’єктами та надсилати зображення з камер спостереження на сервер; у режимі реального часу отримувати повідомлення про виняткові ситуації та відображати повідомлення у користувацькому інтерфейсі. Підсистема написана на JavaScript React і працює за принципом PWA (progressive web app). Окремий веб-сервіс для сповіщення працює з сокетами та інтерактивною картою Leaflet.

В ході розробки:

- проведено аналіз систем інтерактивних карт
- розроблено користувацький інтерфейс підсистеми.
- розроблено кросплатформовий додаток.
- розроблено веб-сервіс для оповіщення користувачів
- розроблена система спостереження за об’єктами.

Ключові слова:

КОМП’ЮТЕРНА СИСТЕМА ЗБОРУ ТА ОБРОБКИ ІНФОРМАЦІЇ, REACT, PWA, JS, UI, GIS, LEAFLET, SERVICE WORKERS.

ANOTATION

The object of development is a subsystem for means of monitoring objects of protection. The main purpose of which is the collection and processing of information on exceptional situations.

The system allows: to register and authorize users in the application; observe objects and send images from surveillance cameras to the server; in real time receive notification of exceptional situations and display messages in the user interface. The subsystem is written in JavaScript React and works on the principle of PWA (progressive web app). A separate Web-based notification service works with sockets and an interactive Leaflet card.

During development:

• An analysis of interactive map systems was conducted

• A subsystem user interface is developed.

• developed a cross-platform application.

• A web-based service is developed for alerting users

• a system for observing objects is developed.

Keywords:

COMPUTER SYSTEM OF INFORMATION COLLECTION AND PROCESSING, REACT, PWA, JS, UI, GIS, LEAFLET, SERVICE WORKERS.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 045440.002 ТЗ	Масштабовані програмні	4		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Масштабовані програмні	1		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045440.004 ПЗ	Масштабовані програмні	50		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА		
Розробив		Місячний І.В			Масштабовані програмні засоби. Підсистема збору та обробки інформації про виняткові ситуації Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашено А.В					
Консульт.						НТУУ «КПІ ім. І. Сікорського» Кафедра СПіСКС Група КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.005 Д1	Масштабовані програмні	1		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Структурна схема проекту			
	А4	ІАЛЦ.045440.006 Д2	Масштабовані програмні	1		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Алгоритм роботи системи			
	А4	ІАЛЦ.045440.007 Д3	Масштабовані програмні	1		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Процес розпізнавання			
			обличчя. Схема алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Масштабовані програмні	1		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Алгоритм роботи системи			
			моніторингу			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Веб-додаток для подання та перевірки мінімальності комбінаційних схем на базі двовходових логічних елементів».

Галузь застосування: дослідження способів оптимізації роботи систем обчислення.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення веб-додатку, що дозволяв би моделювати комбінаційні схеми і перевіряти їх мінімальність.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Можливість для моніторингу об'єктів
- Сповіщення користувачів про виняткові сиутації

5.2. Вимоги до програмного та апаратного забезпечення користувача

- Процесор: MediaTek, Snapdragon, Kirin, Intel, AMD;
- Оперативна пам’ять: 2 Гб;
- Наявність доступу до мережі Internet;
- Операційна система Windows Phone, Android, iOS, Windows, Linux, Mac OS.

5.3. Вимоги до програмного та апаратного забезпечення серверу

- Процесор: Intel, AMD;
- Оперативна пам’ять: 4 Гб;
- Наявність доступу до мережі Internet;
- Операційна система Windows, Linux.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	28.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.004 ПЗ	Масштабовані програмні	50		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Масштабовані програмні	1		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Структурна схема проекту			
	А4	ІАЛЦ.045440.006 Д2	Масштабовані програмні	1		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Процес розпізнавання			
			обличчя. Схема алгоритму			
	А4	ІАЛЦ.045440.007 Д3	Масштабовані програмні	1		
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Алгоритм роботи системи			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розробив		Місячний І.В.			Масштабовані програмні засоби. Підсистема збору та обробки інформації про виняткові ситуації Технічне завдання	Літ.	Аркуш 1 / Аркушів 1
Перевірила		Петрашенко А.В					
Консульт.						НТУУ «КПІ ім. І. Сікорського» Кафедра СПіСКС Група КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.008 Д4	Масштабовані програмні			
			засоби. Підсистема збору			
			та обробки інформації про			
			виняткові ситуації.			
			Алгоритм роботи системи			
			моніторингу			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			
				1		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП	Арк. 2

ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

- ІАЛЦ.045440.005 Д1. Структурна система проекту.
- ІАЛЦ.045440.006 Д2. Процес розпізнавання обличчя
- ІАЛЦ.045440.007 Д3. Алгоритм роботи системи
- ІАЛЦ.045440.008 Д4. Алгоритм роботи системи моніторингу

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

CDN – мережа для доставки контету.

Cookie - невеликий формат даних, які зберігаються в на пристрої користувача. При кожному запити веб-клієнт (найчастіше браузер) надсилає їх до сервера. Мають невелику кількість опцій, такі як: час життя та налаштування безпеки (HttpOnly).

HTML (Hypertext Markup Language) - стандартна мова для розмітки для створення веб-сторінок.

IndexedDB - асинхронне API для зберігання великих об’ємів даних. Реалізована майже повноцінна база даних в браузері.

JSON (JavaScript Object Notation) - це текстовий формат даних між комп'ютерами.

localstorage - персистентний глобальний JavaScript об’єкт, який надає простий синхронний інтерфейс для зберігання даних в браузері. Найчастіше використовується для зберігання даних, які використовуються впродовж однієї сесії користувача.

PWA (Progressive Web Application) – це такий принцип створення веб-додатків, при якому забезпечується робота в режимі офлайн. Такі додатки можуть бути встановлені на дейваси, як нативні програми.

RUM (real user monitoring) – це система для моніторингу швидкості роботи системи на реальних користувачах.

ВСТУП

Своєчасно отримана інформація дозволяє зводити наслідки будь-яких виняткових ситуацій до мінімуму, дозволяє рятувати життя людей та зберігати їх майно. Сьогодні дуже важливо своєчасно отримувати інформацію про надзвичайні ситуації: витік газу, проникнення в будинок або квартиру і тд.

Завдання виявлення і надання інформації про виняткові ситуації можна вирішити за допомогою комплексних систем безпеки. Такі системи забезпечують оперативну передачі інформації органам відповідних служб та організацій. Головним призначенням охоронної системи полягає в своєчасному і гарантованому сповіщенні власників майна та правоохоронних служб. Створення комплексної високонадійної системи може стати одним із вирішенням даної задачі.

Основною метою дипломного проекту є створення гнучкої і масштабованої системи для моніторингу і оповіщення про виняткові ситуації. Для забезпечення високої швидкодії веб-сервісів і відсутність помилкових повідомлень про вторгнення було використано найсучасніше програмне забезпечення. Функціональність, яка надає система, задовольняє потреби користувача на будь-якому з етапів користування охороною системою. У межах одного проекту було реалізовано цілу систему модулів, кожен з яких виконує окремий спектр задач. Система легко розгортається на будь-які з операційних систем, а додатки для моніторингу і картою підтримується будь-яким сучасним веб-браузером. Для основного мобільного додатку було розроблено зручний і адаптивний інтерфейс.

Для забезпечення масштабування системи, потрібно відходити від стандартних патернів та методів розробки програмного забезпечення. В дипломного проекті важливе місце виділяється мікросервісній архітектурі та різним методам оптимізації веб-додатків. Також в рамках дипломного проекту

було використано сучасні технології, які полегшують процес розробки і надають методи для масштабування системи.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Аналоги. Переваги та недоліки

Охоронні системи вже досить давно використовується і вони вже нікого не здивують. Практично у кожному складі, магазині та офісі вже встановлена така система, основним призначенням якої є попередити або запобігти ситуації, в якій буде завдано шкоду людям або матеріальним цінностям. Алгоритм роботи охоронних систем дуже простий. Спочатку аналізується приміщення і в ході аналізу визначаються місця можливих проникнень на об'єкт, найчастіше такими є вікна і двері. Такі місця блокуються за допомогою різноманітних сенсорів, які фіксують звук, рух, зміну світла, тощо. Також у приміщенні встановлюється хаб - це мозок усієї системи, який отримує і обробляє інформацію від сенсорів. У випадку несанкціонованого відкриття дверей, розбиття вікон або проникнення спрацьовує відповідний датчик. Сигнал від сенсора передається до головної системи, вона обробляє сигнал і в залежності від її налаштувань, вмикає звукову сигналізацію або викликає поліцію або службу безпеки.

Також, слід зазначити, що цільова аудиторія потенційних користувачів системою досить не велика. За рахунок використання карти, як єдиного способу оповіщення про виняткові ситуації, тому для використання системи необхідна участь оператора. Проте, методи вирішення такої проблеми існують і вони описані нижче. При аналізі різних рішень від лідерів ринку “Ring” (https://ring.com/) або “AJAX” (https://ajax.systems/). Можна виокремити кілька недоліків, які вирішуються у даному дипломному проекті. Так як, наш додаток використовує камеру, як головний інструмент для моніторингу, тому більш детального розглянемо як її використовують в інших системах.

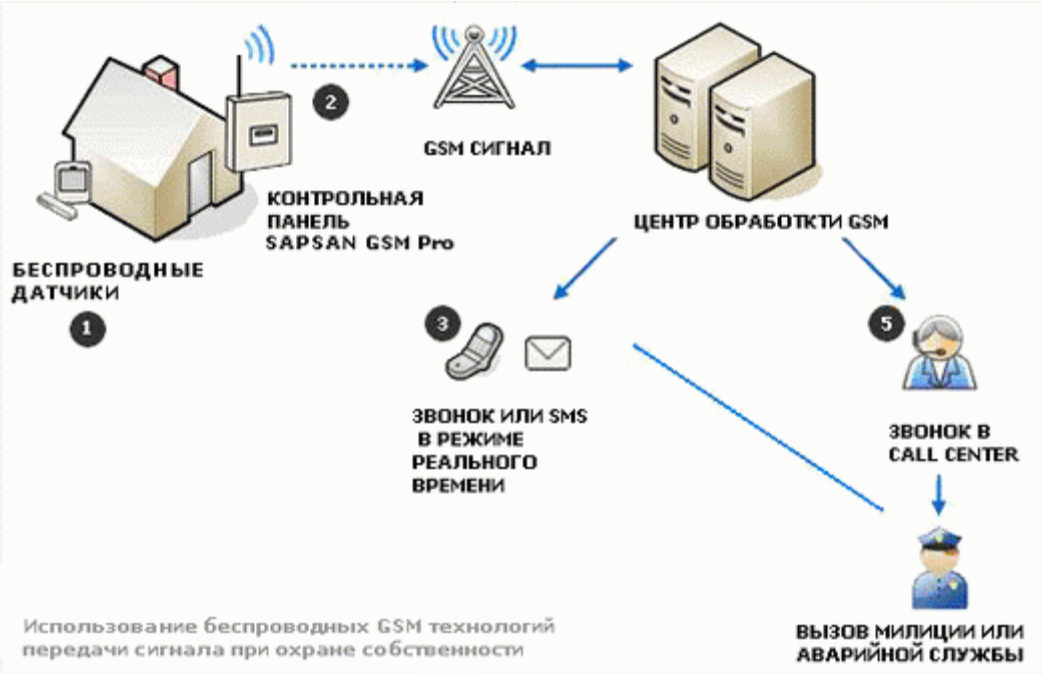

Рисунок 1.1 Принцип роботи охоронних систем

Відсутність можливості використання іншої камери. Найчастіше такі рішення - є монолітними системи. Тому користувач не має можливості використовувати камери від іншого виробника або використовувати інше програмне забезпечення. При проектуванні дипломного проекту в основу системи було закладено мікросервісну архітектуру. Мікросервісна архітектура - це такий архітектурний підхід, при якому кожен модуль системи це окремо існуючий додаток, який має тільки спектр відповідальності. Головною перевагою такого підходу є те, що мікросервіси слабо пов'язані один з одним. Тому, камера, як інструмент збору інформації, був створений, як окремий додаток. Такий підхід гарантує відсутність сильного зв'язку між камерою і хабом системи, тому сенсором для збору інформації може бути будь-який пристрій з камерою. Користувач може сам собі обрати камеру, яка буде задовольняти його потреби і вимоги.

Складність налаштування. Якщо користувач не має технічних навичок і не має змогу самостійно налаштувати систему, то для встановлення системи охорони потрібно запрошувати майстра або команду майстрів, які можуть

провести інсталяцію сенсорів, правильно встановити і налаштувати хаб. З додатком і зручним інтерфейсом складнощі налаштування та час витрачений на цей процес мінімізуються.

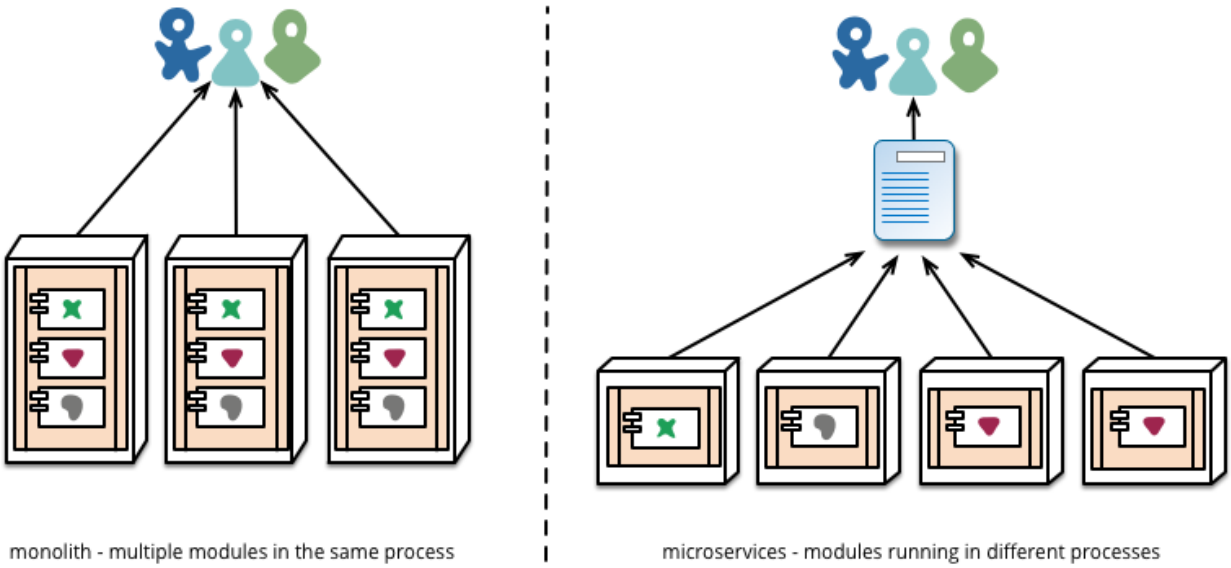

Рисунок 1.2 Структурне зображення монолітних систем і систем на основні мікросервісів

Це досить помітно, коли користувач часто змінює об’єкти моніторингу і не хоче витрачати багато часу на демонтаж і монтаж обладнання.

Звичайні сенсори мають не великий потенціал для програмування, а отже гнучка зміна їх поведінки - це досить нетривіальна задача. Також, досить часто користувач просто забуває вимкнути систему охорони. Користувач відкриває двері, сенсори фіксують рух і вмикається сигналізація. Тобто, сенсор, якщо він працює в режимі моніторингу, спрацьовує завжди, якщо будуть зміни руху. Як було зазначено вище, наша системи складається з кількох підсистем. Ще одним мікросервісом - є сервіс з ідентифікування людей на фото, для цього використовується машинне навчання. Машинне навчання дуже стрімко набирає популярність серед розробників, тому зараз існує велика кількість бібліотек і фреймворків з відкритим сирцевим кодом, однією за таких бібліотек ми і скористувались. Тобто, навіть якщо власника

буде зафіксовано у кадрі, то система зможе його ідентифікувати і не вмикати сигналізацію.

Людський фактор. Коли камера транслює зображення на монітори охоронців або на телефон власника охоронної системи, то в такі моменти визначним стає наявність людського фактору. Зовсім неважливо які технології використовуються для створення системи, якщо людина не помітить факт проникнення до квартири або офісу. На нашу думку, те що може бути автоматизовано, повинно бути автоматизовано. Як було зазначено вище один з сервісів використовує машинне навчання. Сервіс не тільки може ідентифікувати власника, а і ідентифікувати людину, яка не належить до тих, хто не має доступ до об'єкту і сповісти про проникнення. При такому підході до моніторингу, ми мінімізуємо факт помилки людини.

Можливість кастомізації. Так як, більшість охоронних систем - це нероздільний моноліт, тому дуже складно адаптувати їх під потреби кожної людини. Завдяки мікросервісний архітектурі і відсутності сильним зв'язків між модулями, у користувачів є можливість створювати новий модулі, які задовольняє їх потреби і додати його до системи.

1.2 Функціональність, яка може покращити продукт

Проте систему не можна вважати досконалою, вона також має недоліки. Частину з них можна виправити і покращити у майбутньому.

Інформаційна безпека. Мікросервісна архітектура хоч і дає велику кількість переваг і можливостей, які можуть суттєво покращити будь-яку систему, проте існує проблема зі збереженням даних користувачів у безпеці. В поточному стані система має велику вразливість при обміні даними веб-сервісами. Проте існують стандартні і дієві методи для підвищення рівня безпеки і цілісності інформації. При кожному запиті потрібно проводити аутентифікацію користувачів і тільки потім обробляти його.

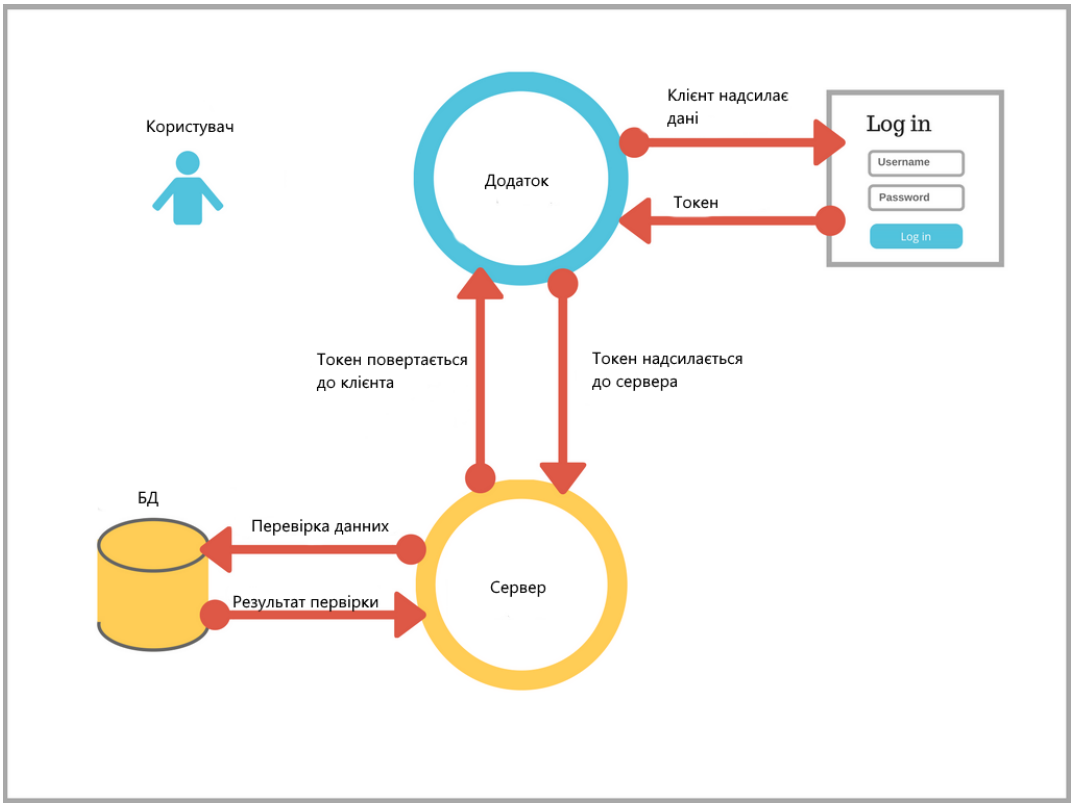

Рисунок 1.3 Схема роботи JWT-токенів.

Для того, щоб підвищити швидкість розробки системи, не було повністю реалізовано системи авторизації й аутентифікації. На даний момент, найпопулярнішою технологією для вирішення цієї проблеми є JWT-токени. Принцип роботи роботи токенів досить простий.

1. Користувач авторизується у додатку. Майже завжди для цього використовується звичайна форма, в якій потрібно вказати свій пароль і почту або просто унікальне ім’я користувача.
2. Якщо користувача було успішно авторизовано, то сервер у відповіді присилає певний токен. Токен - це певний хеш, який має свою структуру і певний алгоритм створення. Наприклад, якщо такий запит зроблено з браузера, то існує декілька варіантів для локального збереження даних. Наприклад localstorage, cookie, indexedDB або експериментальний kvstorage. Проте, стандартним рішенням для заберігання токену є cookie.

3. При кожному запити до сервера токен прикріплюється до тіла запиту.
4. При наявності токена в запиті, сервер перевіряє його валідність і наявність цього токена в базі. Якщо даний токен не є валідним, то сервер відповідає помилкою.

Якщо система побудована за принципом мікросервісної архітектури, то таку перевірку токенів треба робити на кожному з веб-сервісів і ідентифікувати користувача.

Система моніторингу реальної швидкості роботи системи у користувачів. Синтетичне тестування системи в ідеальних умовах важливе для пошуку негативних змін в кодовій базі систем, проте його недостатньо для оцінки швидкості роботу інтерфейсу та веб-сервісів, за якими працює додаток. Для збору таких даних у користувача існує моніторинг швидкості на стороні користувача RUM (RUM – real user monitoring). Для функціонування RUM достатньо підключити одну з систем веб-аналітики (Яндекс.метрика, Google Analytics) і слідкувати за швидкістю завантаження застосунку на пристроях користувача. Взагалі завдяки таким системам можна збирати різну статистику: як користувачі взаємодіють з інтерфейсом, якими пристроями користуються користувачі і тд. За допомогою цих метрик можна постійно змінювати і покращувати систему.

Відсутність налаштувань. Їх відсутність не є якоюсь глобальною проблемою, проте для забезпечення максимально зручної взаємодії із системою вони досить важливі. Гадаю, що декілька налаштувань могли б сильно покращити користувацький досвід. Наприклад, можливість налаштування графіків роботи камери, можливість додавання вже існуючого користувача до іншого облікового запису без проходження реєстрації або можливість записувати хто і у який час був зафіксований на камері.

Відсутність можливостей для аналізу звуку. Вагомою перевагою популярних охоронних систем - це наявність великою кількості сенсорів і

датчиків, одним з таких є датчик звуку. Фіксування розбитого вікна або розмов - є гарним для запобігання проникненню. Так як для фіксування і ідентифікації використовується фото, то система не може отримати і аналізувати звуки, які могли б фіксуватися на камеру.

Повнота екосистеми. Важливим критерієм для вибору охоронної системи - це велика кількість датчиків і сенсорів, які працюють з одним хабом і в межах однієї системи. Тобто, користувачу не потрібно встановлювати дві різні системи охорони, тому що одна з них не має у своєму асортименті сенсорів світла, наприклад. При достатній повноті екосистеми користувач може легко збільшувати кількість сенсорів і все це буде працювати с одним хабом. Проте, варто зазначити, що екосистему дипломного проекту досить легко розширювати і додавати нові модулі. Користувач, якщо він має можливість, може сам створювати свою модулі і доповнювати існуючу систему. Навколо таких систем може створюватися певна спільнота розробників, які будуть розробляти і поширювати самописні модулі для системи. За рахунок зручного інтерфейсу, користувачі можуть легко інтегрувати їх із системою.

Важливим етапом в покращенні даної охоронної системи – це можливість змінювати методи оповіщення про виняткові ситуації. В даному дипломному проекті було реалізовано окрему систему оповіщення і інтерактивну карту, яка відображає стан кожного з сенсорів, які були встановлені користувачем. Такий підхід до інформування користувачів примушує до постійного спостерігання за мапою сенсорів. Для покращення зручності користування системою, гарним покращенням було б додавання можливостей до додавання альтернативних методів оповіщення користувачів. Такими методами можуть стати повідомлення на телефон, пошту або будь-який месенджер. Додавання такої можливості могли б значно розширити і збільшити коло потенційних користувачів системою.

Після аналізу аналогів і готових рішень можна зробити висновок, що система, яка була створена як частина дипломного проекту має свої переваги перед іншими, проте не позбавлена недоліків. Вибір охоронної системи залежить від потреб користувача. Якщо потрібно встановити комплексну систему охорони для жилого будинку, то краще використовувати вже готові рішення. Це зумовлено великою кількістю сенсорів і датчиків, які мають один інтерфейс взаємодії з хабом. Проте, якщо стандартний перелік можливостей, які надають звичайні системи, не можуть вирішити специфічних потреб користувача, то є сенс у використанні таких систем, як була створена для даного дипломного проекту.

2. СТРУКТУРА ПРОГРАМИ

2.1 **Загальне про систему**

Однією з головних проблем великих монолітних програм є масштабування. Це зумовлено тим, що відсутня можливість окремо виконувати певні частини системи. Замість підняття окремого вузла системи, існує необхідність запускати ще одну таку саму систему. Взагалі, виокремлюють два види масштабування, горизонтальне і вертикальне. Вертикальне масштабування – це підняття продуктивності кожного компонента системи з метою збільшення загального рівня ефективності всієї системи. Підняття продуктивності відбувається за рахунок додавання до вузлів більше потужного обладнання. Це самий простий спосіб, так як він не потребує змін в програмному коді вузлів системи. Горизонтальне масштабування має досить велику відміну, від вертикального. Замість того, щоб постійно збільшувати ефективність серверів, які запускають велику і монолітну систему, то при такому підході відбувається розбиття системи на маленькі вузли. Так як, вузол - це окремий і незалежний додаток, то його дуже легко підтримувати і запускати, коли на це є певна необхідність. Замість розгортання ще однієї системи, розгортується тільки маленька її частина. Найпростішим варіантом для реалізації горизонтально масштабування - є дотримання мікросервісної архітектури. Окрім збільшення продуктивності, такий підхід до побудови систем вносить ще декілька позитивних аспектів:

- Суттєво зменшується вартість і час розробки.
- Зменшується час розгортки.
- За рахунок чіткого розумінню границь відповідальності зменшується час на тестування.

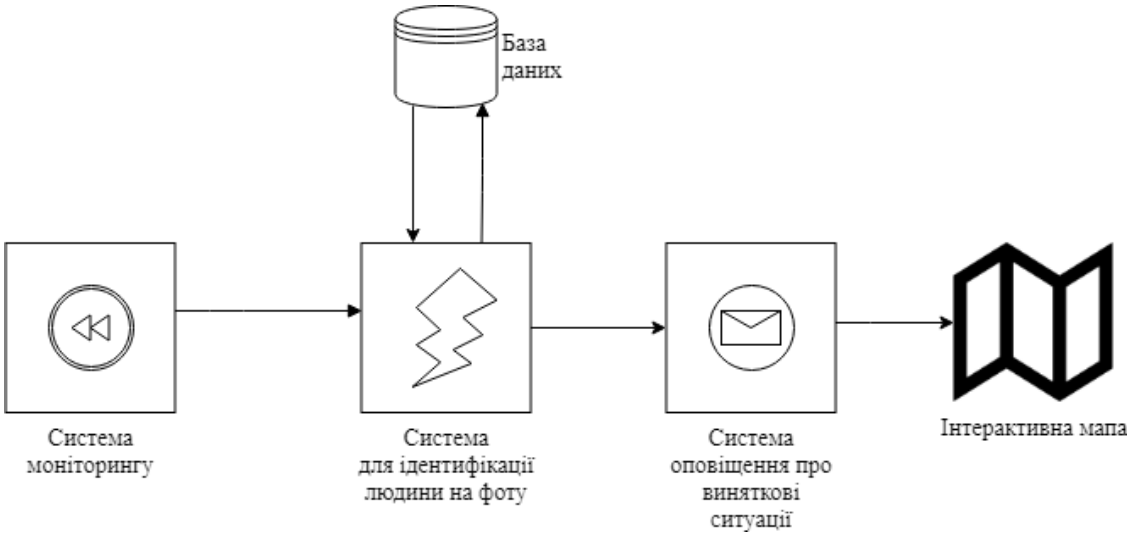

Рисунок 2.1 Схема взаємодії модулів системи.

Проте така архітектура має ряд недоліків з якими можуть зіштовхнутися розробники програмного забезпечення.

Розробка стає значно складнішою. Так як всі компоненти системи, тому потрібно постійно підтримувати однорідність кодової бази, валідації веб-запитів і відповідей на запити. Потрібно обережно обробляти запити між модулями. У випадках, коли один з модулів недоступний, то в модулі, які працюють з ним, дописувати код, який буде оброблювати такі випадки. При інкрементальному збільшенні кодової бази, значно збільшується вартість підтримки.

Існує вірогідність існування декількох баз даних і процес транзакцій між ними можуть бути досить складними.

Підвищується складність тестування таких систем. При використані монолітної системи для тестування необхідно локально запустити одну програму, у випадку мікросервісів необхідно запускати і тримати в актуальному стані все мікросервіси, які необхідно протестувати.

Кожний окремий додаток повинен бути запущений на окремих серверах, що значно підвищує вартість і складність їх підтримки.

2.2 Система моніторингу

Система моніторингу - це поступовий веб-застосунок (PWA), який взаємодіє з системою ідентифікації людей. PWA - це гібрид нативного мобільного додатку та звичайної веб-сторінки. При відсутності мережі, додаток можуть працювати в режимі офлайн. Головними складовими для нього є Service Worker для кешування сирцевого коду на пристрої користувача та маніфест.

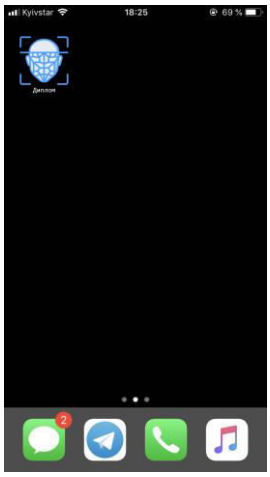

Рисунок 2.2 Встановлений додаток на пристрій користувача

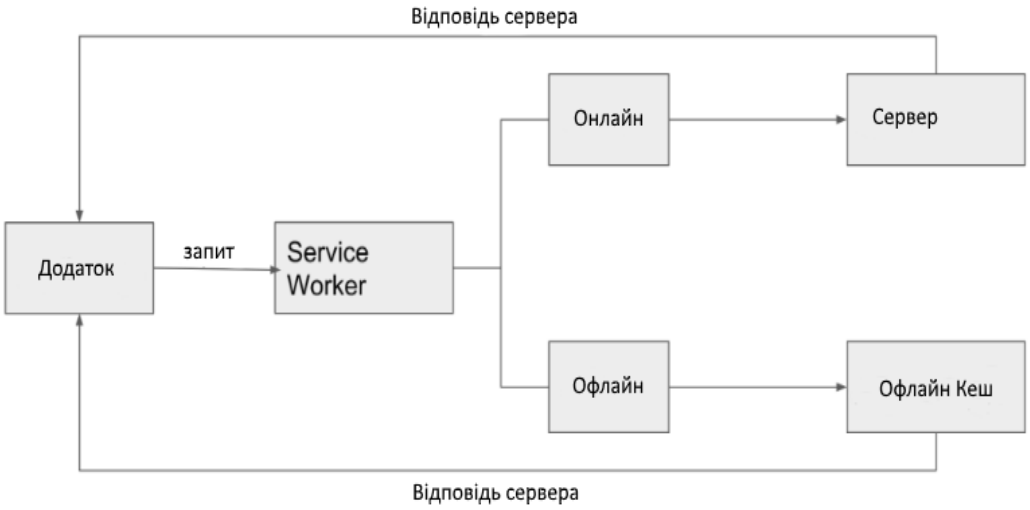

Рис 2.3 Схема реалізації PWA-застосунків.

На рисунку 2.3 зображено процес встановлення додатку на мобільний телефон.

Маніфест - це конфігураційний JSON файл в якому описується метадані, які описують застосунок, наприклад:

- URL, за яким відкривається застосунок.
- URL, за яким будуть доступні іконки.
- Назва додатку.
- Налаштування дисплею, наприклад повноекранний.

Service Workers - це звичайний JavaScript файл, який може модифікувати веб-сторінку, до якої він був підключений. Зазвичай виконує роль звичайного проксі-сервера, який працює між сторінкою і мережею. Коли веб-клієнт робить запит на файл, то воркер зберігає його у кеші браузера. Якщо у користувача відсутній зв'язок з мережею, то файли будуть завантажуватися із кеша. Виконувати роль локального сховища файлів можуть indexedDB або Cache Storage.

Важливим кроком для забезпечення зручного користування додатком - є оптимізація клієнтської частини.

Так як, PWA накладає обмеження на розміри файлів, тому дуже важливо правильно мінімізувати файли, які будуть потрапляти до Service Worker. Найпоширенішим рішенням для цієї задачі - є використання різноманітних бандлерів. Бандлер - це система, яка обробляє і мініфікує статичні файли. За допомогою великої кількості вже готових плагінів, можна суттєво зменшити розмір .js файлів і зображень. Справа в тому, що в завантаження JavaScript файлів, збільшує час до першої взаємодії користувача з додатком, а якщо таких файлів декілька, то користувач може не дочекатися контент і закрити вкладку. Тому для покращення продуктивності бандлери збирають декілька JavaScript файлів в один. На даний момент, браузери працюють таким чином, що завантажити і проаналізувати один великий файл швидше, ніж декілька

маленьких. Найпопулярнішим інструментом для створення бандлу, який є стандартом серед інструментів – WebPack.

Підтримка старих браузерів. Так як, на ринку існує велика кількість браузерів і кожний з них може не підтримувати деяки можливості новітнього JavaScript, тому для збільшення потенціальної кількості користувачів важливим кроком - є використання різноманітних трансляторів коду. Головним завдання якого, трансформування сучасного коду в такий, який буде підтримуватися будь-яким браузером. Найпопулярнішим інструментом для транслювання коду є Babel. Його також можна використовувати як плагін для бандлера чи окрему CLI-утиліту.

Рис 2.4 Схема роботи Babel.

Ще одним важливим кроком для оптимізації застосунку - є мініфікація коду.

Мініфікація коду - це процес спрямований на зменшення розміру початкового коду, шляхом видалення непотрібних символів та конструкцій не змінюючи початкову функціональність коду. Одним із найпопулярніших мініфікаторів коду є uglifyJs, який вміє зменшувати код більше ніж на 40% від початкового розміру файлів.

Звичайно, для створення мобільного додатку можна було використовувати нативні технології, наприклад: мови Swift для IOS і Java для Android, програмування на яких надає кілька переваг:

По-перше ОС гарно оптимізують нативні рішення, тому ефективність програми була б вище, ніж веб-сторінка в браузері.

По-друге була б можливість повноцінного доступу до API пристрою. При використанні якого, значно би покращилась взаємодія з камерою.

По-третє використання PWA накладає деякі обмеження на розміри застосунку. Наприклад, саме це обмеження зумовло створенню окремого додатку для відображення сенсорів на інтерактивній мапі.

Проте, у веб-застосунку є кілька переваг перед нативним рішенням.

По-перше, такий додаток - майже гарантує одну кодову базу для всіх операційних систем та пристроїв. Тобто, не має потреби у створенні декількох однакових додатків на різних мовах програмування, щоб задовольнити потреби всіх користувач, які використовують різні ОС.

По-друге, відсутність взаємодій з різними маркетами (Apple Store та Android Market, наприклад) додатків.

Проаналізувавши застосунок, було зроблено висновки, що в ньому не будуть виконуватись складні обчислення, тому якщо у нативної і є певна перевага у продуктивності, то вона мінімальна. Останні декілька років веб-технології досить потужно розвивається, тому програмний інтерфейс, який надає сучасний браузер, досить часто вистачає, щоб задовольнити потреби користувачів.

Функціональність, яку має додаток:

- Реєстрація нових користувачів.
- Авторизація користувачів.
- Режим моніторингу з надсилання зображень на сервер.

2.3 **Система ідентифікування**

Система для ідентифікування - це основний модуль системи, яка є з’єднує систему моніторинга та оповіщення. Модуль отримує дані від додатку, аналізує їх, а результати аналізу зберагіються у базу даних. Потім вони використовуються авторизації та ідентифікації користувачів. Працює на

JavaScript платформі NodeJS. Якщо користувач має специфічні потреби має можливість створити свій вузол, то саме цей мікросервіс стане точкою входу до всієї системи. Вся взаємодія з модулем відбувається за звичайним REST інтерфейсом. Якщо існує необхідність в нових модулях, то можна просто розширювати цей інтерфейс для роботи з новими сутностями. REST архітерктура також позитивно впливає на можливості системи до масштабування, а саме:

- Простота стандартного інтерфейсу.
- Відкритість компонентів до змін функціональності для задоволення потреб користувача.
- Прозорий і не складний зв’язок між модулями системи.
- Відсутність станів

Ідентифікація людей за фото відбувається завдяки використанню бібліотеки для машинного навчання face-recognition.js. Ця бібліотека є обгорткою для мови JavaScript, над іншою популярною бібліотекою OpenCV. Обгортка має більш просте API та простий процес інсталювання.

2.4 **Система оповіщення**

Система оповіщення - модуль системи, яка відповідає за сповіщення про фіксацію камерою невідомої людини. Модуль також працює на платформі NodeJS, проте не має REST-інтерфейсу для взаємодії з ним. Замість звичайних HTTP-запитів, система обмінюється повідомлення за допомогою WebSocket. У разі фіксації несанкціонованого доступу, сервіс буде надсилати повідомлення до інтерактивної карти користувача. Використання сокетів зумовлено потребою в оновленні мапи в режимі реального часу. Альтернативним вирішенням цієї задачі - є звичайний HTTP-запит з певним інтервалом, поллінг. Як було зазначено вище HTTP запити на мають стану, для помітки користувачів використовуються різноманітні маркери: хедери, cookie та сесія, які передаються з кожним новим запитом. Такий підхід значно полегшує масштабування системи, тому що не потрібно копіювати і

переносити стан кожного користувача. Проте для вирішення задач, які повинні оновлюватися в режимі реального часу, такі запити тільки створюють зайве навантаження. З ростом користувачів і відкритих мапах, в яких відбувається поллінг між браузером і сервером, сервера можуть не витримувати такого навантаження, що погано позначиться на роботі всієї системи. Тому, для підвищення продуктивності вибір зупинився на сокетах.

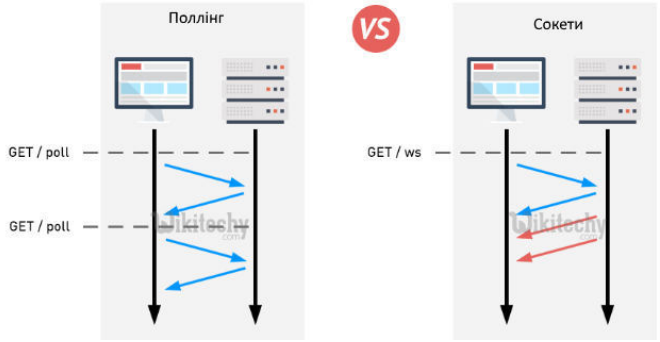

Рисунок 2.5 Відмінність полінгу від сокетів.

Варто зазначити, що масштабування сокетів організовується декілька важче, ніж звичайний REST підхід. Сценарії використання веб-сокетів умовно можна розділити на два основні випадки. Перший видок, це коли для з'єднання з сервером користувач авторизується в системі. Сам процес авторизації відбувається так само, як і при звичайному HTTP-запиті. Відправка даних у хедері запиту на сервер, у разі їх правильності сервер відкриває з'єднання для сокету. Для підвищення продуктивності і надійності системи балансувальник навантаження може запускати декілька однакових додатків на різних серверах, тому наявність стану, при використанні сокетів, створює необхідність в обміну даними між серверами. Гарним прикладом такого стану – це авторизація користувачів. Сам факт авторизації і активного сокету зберігається в оперативній пам'яті сервера і отримати доступ до неї з іншого об'єкту додатку неможливо. Тому, для вирішення проблеми з обміном даними, використовується база даних, в якій зберігається вся потрібна інформація про сокети. В ролі буферної бази даних може використовуватися

будь-яка з існуючих баз даних, проте найчастіше використовується Redis. Redis – це швидке сховище для структур «ключ-значення». Всі дані такого сховища зберігаються в оперативній пам’яті. В умовах використання такого підходу до зберігання інформації, сховище не потребує доступу до дискових накопичувачів, що значно підвищує швидкість пошуку даних. При використані буфера, вся інформація про з’єднання буде зберігатися в базі даних і будь-який сервере, який запускає додаток, має доступ для цих даних. В другому випадку, при відсутності наявності авторизації для користувачів, відсутня проблеми з обміном станами між серверами. Також, важливою проблемою використання сокетів – це підтримка актуальних даних для всіх з’єднань. Наприклад, існує певний сайт, який реалізує функцію звичайного чату. Чат працює в режимі реального часу і повинен оновлюватися кожен раз, коли будь-який учасник бесіди надішле повідомлення. Нехай, чатом користується дві людини, перша людина має назву сокета socketA, а друга socketB. Повідомлення користувача надсилається до сервера по сокету socketA і там оброблюється. Як повідомити socket, що сервер отримав повідомлення? Існує декілька варіантів. Найлегший – це повідомити всі сокети, які з’єднані з сервером про те, що він отримав повідомлення. Це досить гарне і просте рішення до тих пір, коли сервер має малу кількість з’єднань і не потрібно виконувати велику кількість обчислювальної потужності для цього. Тому, для підвищення продуктивності системи, використовується Redis, про якого було зазначено вище. Основний принцип оповіщення інших сокетів про отримання повідомлення – це використання простого і поширеного патерна проектування «публікація-підписка». Алгоритм патерну дуже простий і гарно підходить для вирішення проблему з оповіщенням. Кожний сокет при з’єднанні з сервером підписується на оновлення бази даних. Коли сервер отримує повідомлення від socketA, він робить новий запит в базу даних і записує це повідомлення в неї. Після успішного запису даних, всі сокети, які були підписані на зміни бази, отримають повідомлення про те, що вона оновилася і сокети завантажують

нові дані. Таким чином, все навантаження розподіляється між основним сервером і базою даних.

2.5 **Інтерактивна мапа**

Інтерактивна мапа - це окремий веб-додаток, головною задачею якого, є сповіщення користувача про виняткову ситуацію. Для відображення мапи використовується бібліотека Leaflet, яка має відкрити серцевий код. Між браузером користувача і системою оповіщення постійно є активне підключення і у разі отримання повідомлення від сервера, то на карті можна спостерігати камеру, яка зафіксувала людину. Звичайно, мапу можна було не виокремлювати в окремий додаток, а просто додати його до PWA-застосунку для системи моніторингу. Проте, є проблема з різною поведінкою у браузерах. Сучасні браузери накладають певні обмеження на розміри файлів, які повинні зберігатися на пристроях користувача, а інтерактивні мапи займають досить великий об’єм . І кожному браузері цей поріг відрізняється від іншого, наприклад в Google Chrome - це 6% від вільної пам'яті, а в Safari 50 мегабайт. Тому, для створення однакової поведінки в усіх браузерах, було прийнято рішення про створення окремого застосунку. Проте, при розробці додатку виявилась відсутність можливості обмінюватись даними користувача, який вже авторизувався у додатку для моніторингу. Це пов’язано з тим, що при переході на додаток з мапой сторінка браузера перезавантажується і дані, які були збережені втрачаються. На даний момент, основним критерієм для авторизації користувача є його username. Одним з методів вирішення задачі є - додати у додаток ще одну форму авторизації. Проте, було відкинутого такий варіант, так як при збільшенні кроків для авторизації повторне проходження авторизації стане незручним користувацьким досвідом. Для того, щоб зробити використання системи максимально зручним, було реалізовано наступне. Два веб-додатки (система моніторингу і мапа), є різними модулями, які розгортаються на серверах незалежно один від одного, тоді було вирішено

розгорнути два застосунки так, щоб вони мали однаковий URL. Проблему було вирішено за допомогою використання Ngnix. Ngnix - це такий веб-сервер, який використовується як зворотній проксі-сервер. Зворотній проксі-сервер - це тип проксі-сервера, який ретранслює запити з веб-клієнтів на один або декілька серверів, які розташовані у внутрішній мережі. Також Ngnix може відігравати важливу роль у спроможності системи до горизонтального масштабування, він дозволяє описувати групи серверів в конфігураційних файлах і балансувати навантаження між ними за допомогою певних алгоритмів, які закладено в систему. Є можливість задавати «вагу» для серверів. Така характеристика є досить важливою для балансування системи. Справа в тому, що якщо сервер використовує алгоритм Round-Robin, то Ngnix буде рівномірно розподіляти навантаження але при цьому будуть враховуватися вага кожного з серверів. Наприклад, локально запущено 8 процесів нашого застосунку. Кожний окремий процес працює з певним портом, який був виділений під нього.

```
upstream backend {
      least_conn;
      server 127.0.0.1:8000;
      server 127.0.0.1:8001;
      server 127.0.0.1:8002;
      server 127.0.0.1:8003;
      server 127.0.0.1:8004;
      server 127.0.0.1:8005;
      server 127.0.0.1:8006;
      server 127.0.0.1:8007;
}

server {
        listen 0.0.0.0:80;

        location / {
                proxy_pass http://backend;
        }
}
```

Тепер Nginx буде приймати запити на 80 порт і балансувати їх між процесами додатку. При цьому, запит буде надходити до тих серверів, які в даний момент найменше завантажений. Nginx створює процеси-воркери, кожен з яких може обслуговувати тисячі запитів. Воркери досягають такого

результату завдяки механізму, який було створено на основі швидкого циклку. Розділення осоновної роботи від обробки запитів дозволяє кожному з воркерів виконувати обчислювальні дії і переривати їх тільки після отримання події по з'єднання. Кожне з'єднання оброблюється воркером і переміщається в event-loop. В цьому циклі події оброблюються асинхронно і дозволяючи оброблювати задачі в неблокуючій манері. Такий підхід дозволяє гарно масштабуватися і підвищувати продуктивність системи.

Таким чином, коли користувач потрапляє на додаток з картою, то запит буде проходити через зворотній-проксі север, який поверне звичайний HTML-документ. При такому підході, контент для користувача буде змінюватися, проте URL сторінки завжди буде залишатися незмінним. Залишилось зберігати username в будь-якому персистемному сховище даних. Наприклад, localStorage або cookie.

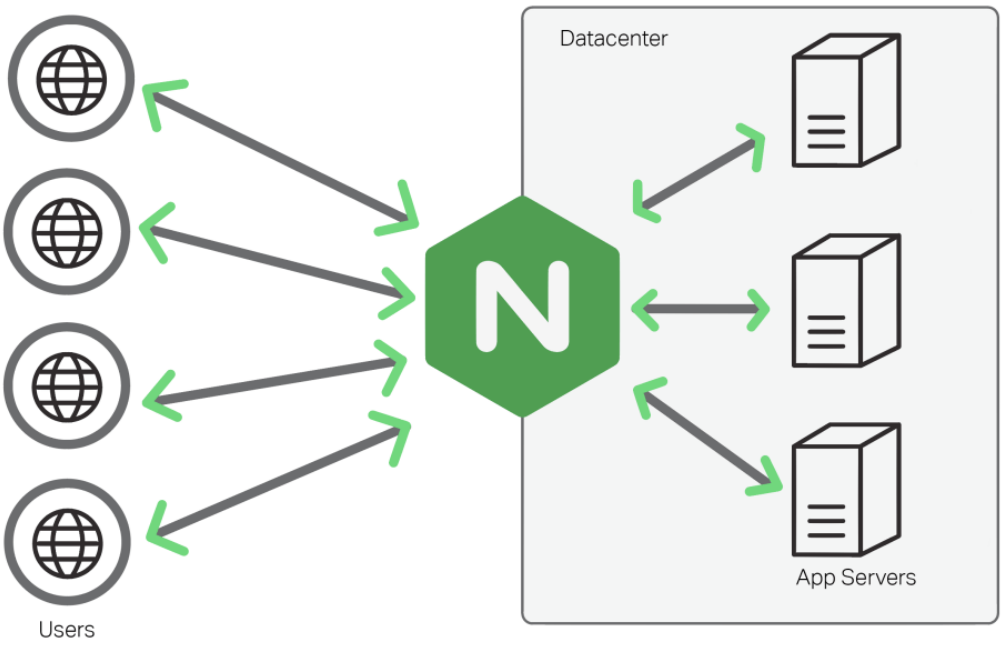

Рис 2.6 Схема роботи Ngnix серверів.

Ще одним важливим кроком у підвищенні продуктивності і масштабованості системи – це використання CDN (Content Delivery Network). Найчастіше такою мережею є множина спеціалізованих серверів з

спеціалізованим програмним забезпеченням, які прискорюють доставку (отримання даних користувачем) даних. Сервера розташовані по всьому світу таким чином, щоб час відповіді від сервера був мінімальним. Такими даними найчастіше є статичні елементи веб-сайтів. Окрім зображень, тексту, відео такими даними можуть юути досить неочікувані речі, наприклад: ігри (сам код ігор поширюється клієнтам за допомогою мережі) або оновлення для операційної системи. Взагалі весь контент, який отримує клієнт можна розділити на дві категорії.

Динамічний контент формується в момент отримання запиту від користувача. Для формування найчастіше використовується бази даних.

Статичний контент на сервері знаходиться вже у готовому вигляді і для його створення не потрібно виконувати запити до бази даних або виконання інших скриптів. Тому, хто б не робив запит на отримання такого контент, то сервер завжди буде повертати одне і теж. Важливо, що від запиту до запиту вміст не буде змінюватися.

Статичний і динамічний контент створюють різний тип навантаження. У випадок коли користувач робить запит на отримання динамічного контенту, то важливим фактором швидкодії є процесор, I/O (для баз даних) і кількість пам’яті. У випадку коли користувачу потрібен статичні файли, то процесор майже не грає ніякої ролі, I/O важливий тільки для тих файлів, які не були закешовані, найважливішим фактором – стає швидкість мережі. Взагалі, можна зробити так, щоб сервер надсилав користувачам і динаміку і статику, проте в такому випадку є декілька негативних аспектів. Виконання двох функцій – це суміщення ролей, які заважають один одному. Особливо велике навантаження відбувається в моменти, коли IO від стики починає заважати IO від динаміки. Ще однією важливою деталлю є те, що контент, який формується динамічно, найчастіше потребує зберігання певного стану користувача на сервері (сесія і пов’язані з нею дані). А статичні дані його не

потребують. Саме в умовах відсутності стану, який потребує складної двосторонньої синхронізації. Тому сервери, які доставляють статичні дані, мають здібність до горизонтального масштабування. Великі компанії, які мають високонавантажені продукти надсилають статичні файли за допомогою CDN. Завдяки використання такої технології, значно зменшується навантаження на веб-сервери з динамічним контентом, за рахунок перенесення за них функції доставки статичних даних.

В основі роботи систем CDN працює протокол прикладного рівня BGR. Користувач робить запит для отримання HTML-файлу від сервера, а весь статичний контент з системи доставки контенту, яка розташована на піддомені основного сайту. Наприклад, основний сайт доступний за адресою www.diplom-app.com, а CDN www.cdn.diplom-app.com. Коли веб-браузер виконує звертається за цією адресою (на адресу з CDN), то завдяки протоколу BGR його запит надсилається до найближчого вузлу мережі. Маршрутизатор провайдера обирає найближчий. В результаті чого, запит відправляється на найближчий веб-сервер, який відповідає на запит аналогічно по найкоротшому шляху. Зі сторони власників сайту існує декілька варіантів для наповнення мереж контентом:

- Файли завантажуються в сховище будь-яким зручним способом, наприклад, FTP або SCP. В сховищі або в спеціальній панелі для керування їм, визначається dns-ім’я, за яким можна завантажити ці файли. Саме ця адреса вказується в основному HTML-файлі сайту.
- Користувач визначає для домена ‘origin’ заголовок, після чого, за запитом клієнта, CDN завантажує файли з сайту, до якого вона підключена, і надсилає їх до браузера користувача.

Таким чином весь статичний контент завантажується значно швидше, ніж HTML-сторінка. У такому випадку, коли весь контент завантажено раніше від розмітки, сам процес відтворення сайту не буде блокуватися запитами.

CDN також може вирішити проблеми систем, які мають дуже великий об’єм статичних даних, які надсилаються користувачам. Вартість трафіку CDN нижча. Створення окремою мережі для статичних даних дешевша, ніж апаратне покращення серверів. Проте, такі мережі мають ряд недоліків.

У випадку, коли об’єкт, який завантажує користувач, не знаходиться в кеші вузла мережі. Наприклад, це перший запит на цей файл або його не можна кешувати (HTML-документи, наприклад). В такому випадку користувач отримає додаткову затримку між вузлом CDN і нашим веб-сервером.

Мережі доставки контенту – це складні системи, в яких також можливі збої, помилки та нестабільність. Використовуючи CDN для своїх системи, ми додаємо ще одним рівень складності.

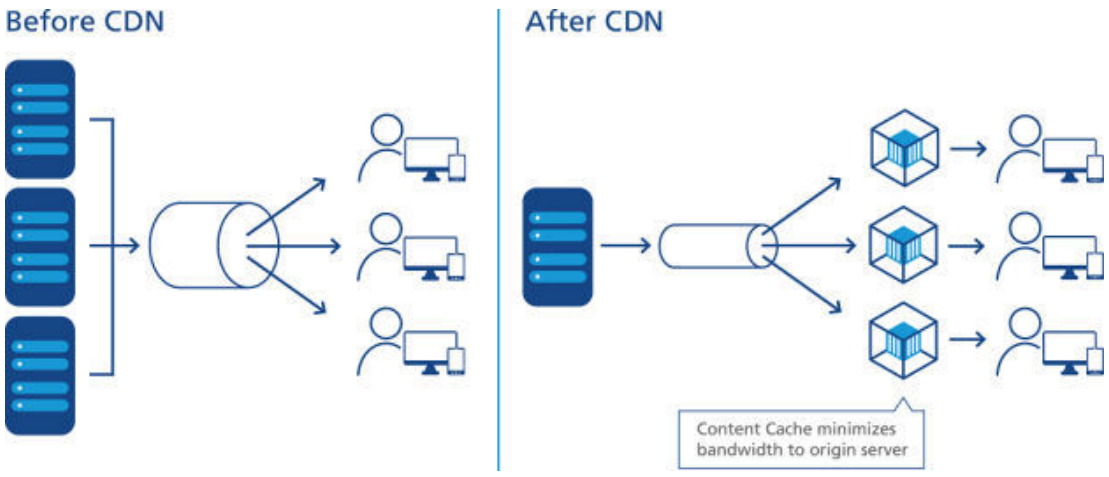

Рисунок 2.7 Структурна схема роботи CDN

При використанні різноманітних технологій і добре закладеної мікросервісній архітектурі можна створити систему, яку можна масштабувати на будь-якому з етапів її роботи. Завдяки розподіленню областей відповідальності між веб-сервісами покращились не тільки можливість системи до масштабування, а і покращилась підтримка системи, що досить сильно впливає на швидкість розробки нових модулів для системи.

3. АЛГОРИТМ РОБОТИ МОДУЛІВ

3.1 Загальний алгоритм взаємодії модулів системи

Розроблений мобільний додаток має зручний і інтуітивно зрозумілий інтерфейс. Взаємодія з додатком здійснюється через декілька екранів. Кожний екран – це окремі частини додатку, які надають можливість взаємодіяти з кожною частиною функціональності системи. Для того, щоб переходити від одного екрану до іншого з самого верху кожної сторінки є блок (далі хедер), на якому розташовані посилання на них. Цей блок, має два формати відображення контенту в залежності від розмірів екрану користувача, звичайний і для мобільних телефонів. Якщо пристрій користувача менше ніж 700 пікселів, то хедер відображається в мобільному режимі. В мобільному режимі хедер зменшується і всі посилання зникають. Для того, щоб посилання

знову відобразились на екрані, необхідно натиснути на кнопку з трьома лініями. Після натискання з’явиться контекстне меню з контентом.

Перше, що бачить користувач, це екран реєстрації нових користувачів.

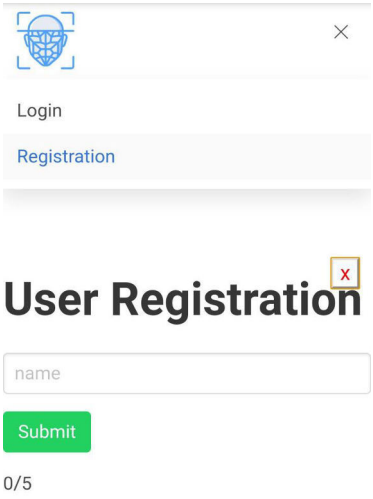

Рисунок 3.1 Сторінка реєстрації корстувачів

В контекстному меню, користувач може обрати розділ, що його цікавить і перейти до нього. Окрім розмірів пристроїв, на відображення посилань в меню впливає факт авторизації користувача у додатку. Наприклад, якщо користувач щойно відкрив додаток, то він зможе перейти до наступних екранів: Login, Registration , а якщо користувач вже авторизувався в системі, то він отримує доступ до Map, Detector і також з’являється кнопка за допомогою якою, користувач має змогу вийти із системи.

Рисунок 3.2 Стан заголовку до авторизації користувача

Рисунок 3.3 Стан заголовку після авторизації користувача

- Login – екран на якому користувач може авторизуватися в системі.
- Registration – екран з реєстрацією, на якій користувач створює в системі свій username і надсилає фото для наступного аналізу.
- Map – інтерактивна карта, на якій зображено всі працюючі сенсори користувача. Слід зазначити, що карта – це окремий додаток і вона не є частиною мобільного додатку для моніторингу.
- Detector – частина додатку за допомогою якого користувач може керувати процесом моніторингу за об’єктами.
- Logout – кнопка за допомогою якої користувач має змогу увійти в систему під іншим користувачем.

3.2 Опис модулів

Додаток для моніторингу складається з декількох екранів, кожен з яких має свою функціональність.

Сторінка реєстрації - являє собою інтерфейс для створення зображень користувача і поле, у якому користувач вказує свій username. Перед відправкою даних виконується валідація форми. Валідація - це процес перевірки даних на їх відповідність з тим, які дані очікує отримати сервер. В контексті сами цієї форми, то тут всього два критерії, щоб форма вважалась валідною і користувач мав можливість відправити її не сервер. Такими критеріями є кількість зображень, які зробив користувач (їх повинно бути п’ять) і довжина username повинна бути більше, ніж три символи. У разі коли дані не валідні, то відбувається блокування кнопки, яка надсилає форму. Якщо

реєстрація відбулась успішно, ім’я користувача зберігаються у cookie браузера.

Сторінка з камерою для моніторингу об’єктів охорони. Моніторинг відбувається за рахунок відправки зображень на сервер з інтервалом у дві секунди та рекурсивною функції. При реалізації цієї частини функціональності, я зіштовхнувся з проблемою. Справа в тому, що для полегшення взаємодії з API камери використовується бібліотека react-html5-camera і вона не дає можливостей для того, автоматичного створення фотографії. Для того, щоб мати можливість програмно викликати функцію для створення фото, потрібно було розширювати стандарте API бібліотеки і додавати функції зворотного виклику, які повертають об’єкт класу камери. Після того як було зроблено фото, воно тимчасово зберігається у пам’яті пристрою у вигляді base64 строки і відправляється вже на сервер. Після успішної доставки фотографії на сервер, пам’ять звільняється.

Рис 3.4 Алгоритм роботи сторінки авторизації.

Рис 3.5 Алгоритм роботи детектора.

Система ідентифікації виконує роль посередника між мікросервісами і обєднує їх в одну єдину систему. Мікросервіс реалізує REST-інтерфейс для передачі даних. Для кожного роуту було створено окремі сервіси, які реалізують функціональність реєстрації та моніторингу.

Для реєстрації нових користувачів використовується `POST /api/user`. Обов'язковими параметрами для роута є ім’я користувача `name`, та масив фотографій `photos`. Після отримання даних від користувачів, відбувається перевірка на наявність даного користувача в базі даних. У разі відсутності користувача, за допомогою FaceRecognition.js на базі фотографій користувача створюється модель, яка записується в базу даних. Модель - json файл, який створюється на основі унікальних дескрипторів обличчя.

Рис 3.6 Алгоритм роботи реєстрації.

В режимі моніторингу додаток надсилає дані на інший URL. POST /api/photo. Обов'язкові параметрами для роута - ім’я користувача name, та та фотознімок з місця моніторингу photo. Після відбувається аналіз зображенням. Спочатку з бази даних завантажують модель користувача, яка була створена ще на етапі реєстрації і за допомогою бібліотеки для аналізу зображень ідентифікується людина. У разі, якщо на фотографії було зафіксовано людину і вона не є зареєстрованою у системі, то надсилається запит на /api/alarm/, цей роут належить системі оповіщення.

Рис 3.7 Алгоритм роботи ідентифікації

Алгоритм роботи FaceRecognition.js.

Для ідентифікації обличчя на зображенні бібліотека OpenCV використовує алгоритм Віоли-Джонса. Алгоритм використовує метод ковзного вікна. Метод базується на використання деякої рамки, меншою від початкового зображення, яка рухається по фотографії і за допомогою системи класифікаторів визначає наявність обличчя.

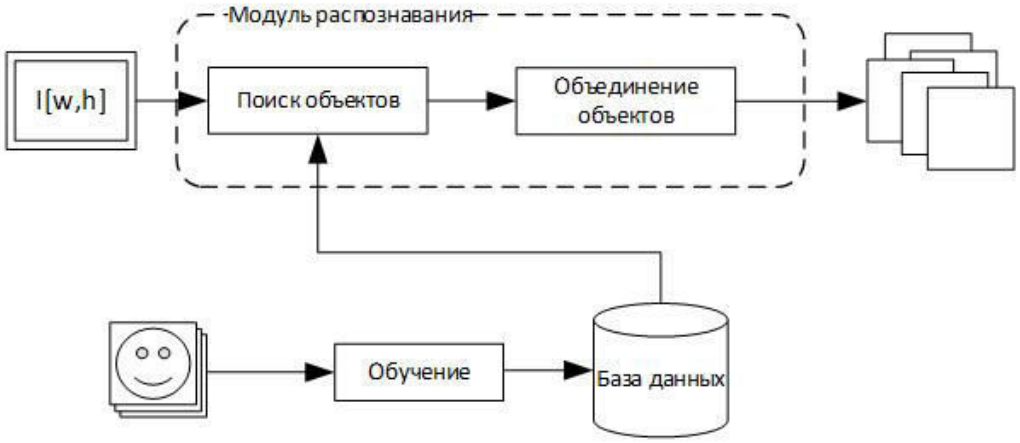

Рисунок 3.8 Алгоритм роботи модуля ідентифікації

Слід зазначити , що система класів базується на евристичній ознаці Хаара. Майже на всіх фотографіях область, де розташовані очі, темніша за область, яка розташована нижче – щоки так ніс. На перший погляд, такі признаки не мають нічого спільного з обличчям людей, проте при всі своїй примітивності вони мають високу узагальнюючу силу.

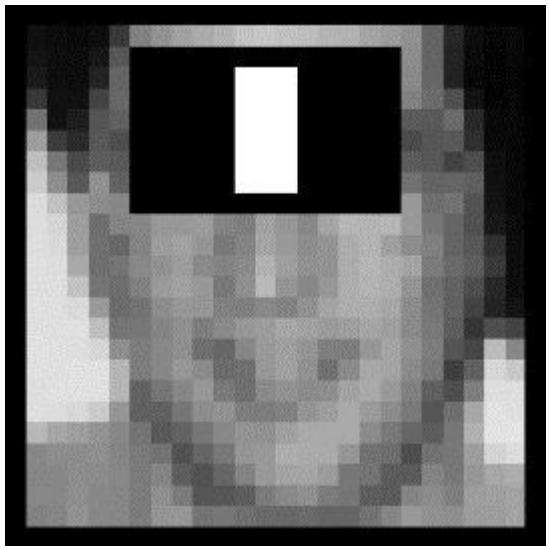

Рисунок 3.9 Ознака Хаара

Пошук обличчя на фото відбувається в декілька етапів:

1. Видалення кольору з зображення і перетворення його в матрицю яскравості.
2. Поверх зображення накладається маска. Маска – це квадратне зображення, яке є ознакою Хаара. Область с ознакою переміщується по всьому зображенню і аналізує його.
3. Проводиться підрахунок даних отриманих в результаті аналізу зображення. Підсумовується значення яскравості тих комірок, які потрапили в білу частину маски і з них вираховується ті, які потрапили в чорну частину. Значення ознаки описується формулою $\boldsymbol{F = X - Y}$, у якій X – сума значень яскравості комірок на білій частині, а Y – це сума значень з комірок на чорній стороні. Якщо хоча б в одному випадку різниця між білими і чорними

частинами перевищує деякий поріг, то ця область зображення зберігається для подальшої роботи.

4. Алгоритм повторюється за другого пункту але вже з новою маскою і тільки з тією частиною зображення, яка пройшла перевірку.

В алгоритмі, який описано вище не вказано про дуже суттєву оптимізацію, яка забезпечую швидку роботу програми. Для того, щоб вирахувати яскравість однієї частини зображення з іншої, то потрібно було б обробляти кожний піксель зображення, тому для уникнення настільки важких обчислень зображення представляється в інтегральному вигляді. Інтегральне представлення зображення можна представити у вигляді звичайної матриці. Розмір такої матриці такий самий, як і у початкового зображення. Кожний елемент такої матриці розраховується за формулою

$$II_{(x,y)} = \sum_{i=0,j=0}^{i \le x,\ i \le y} I(r,c)$$

де $I_{(r,c)}$ – яскравість пікселя початкового зображення. Кожний елемент матриці $II_{[x,y]}$ – це сума всіх пікселів, які містяться в прямокутнику от $(0,0)$ до (x,y). Для того, щоб вирахувати суму прямокутної області в інтегральному представлені, потрібно зробити всього 3 операції. Це дозволяє швидко розраховувати ознаки Хаара для зображення.

Останній етап роботи модуля – ідентифікація. Насправді, це найпростіший крок. Основний принцип зводиться до того, щоб порівняти схожість отриманих ознак з тими, що вже містяться в базі даних.

Проте алгоритм Віола-Джонса має декілька недоліків.

- Досить довгий час роботи алгоритму для навчання. Для того, щоб навчити систему розпізнавати зображення, потрібно проаналізувати велику кількість зображень
- Велика кількість близьких результатів, за рахунок використання технології ковзного вікна.

Інтерактивна карта з відображенням активних сенсорів користувачів. Карта створена за допомогою бібліотеки Leaflet, яка де-факто є стандартом для задач по відображенню карт в браузері, вона має досить добру підтримку сучасними пристроями і досить потужне API, частину якого було використано для додатку з інтерактивною картою.

Тайли. В момент ініціалізації додатку створюється полотно, на якому розміщено базовий набір користувацьких елементів і відсутнє зображення самої карти. За допомогою тайлів можна накладати зображення карт на полотно. Leaflet має досить велику кількість стандартних мап. Географічні, демографічні, топографічні і тд.

Маркери – ледь не основне, що потрібно на інтерактивних картах. Це кольорова іконка на мапі, яка символізує сенсор користувача. Важливими можливостями маркерів є створення кастомізованих іконок і додавання обробників користувацьких подій. Хоч ця функціональність не була використана в рамках додатку, проте Leaflet надає можливість створювати візуальну кластеризацію. Кластеризація – це спосіб відображення маркерів, при якому їх кількість залежить від масштабу карти. Такий спосіб гарно підходить для відображення великої кількості даних.

Все на карті побудовано і тримається на шарах – маркери, точки, кластери, фігури і тд. Можна створювати безліч шарів і розміщати на ньому контент. Також можна додавати і видаляти шари по мірі необхідності розробника.

Leaflet надає досить обширне API для використання користувацьких подій у браузері. Події можуть використовуватися для відстеження кліків користувача на карту або на ішний будь-який об’єкт в документі.

Елементи для взаємодією з картою. У лівому кути карти можна спостерігати кнопку, на якій зображено плюс і мінус, призначенням яких є збільшенні і зменшення масштабу карти. Такі елементи також створює і надає бібліотека.

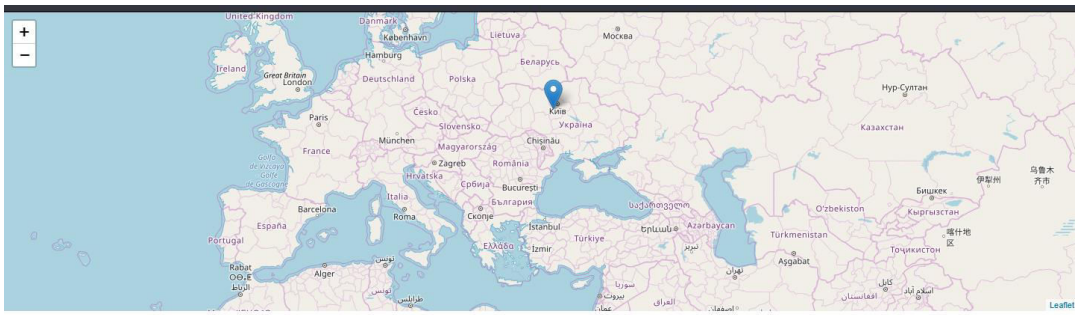

Рисунок 3.10 Інтерактивна карта

Для обміну даними система оповіщення використовує два протоколи: HTTP та WebSocket. Додаток з інтерактивною картою завжди має активне з’єднання з системою оповіщення. При отриманні повідомлення по сокету, точки на карті перемальовуються і у разі винятковою ситуації змінюється стасути сенсорів, які зображені на карті.

Для ідентифікації користувачів використовується застосунок для моніторингу об'єктів. Як було описано вище, проблема з обміном інформації про статус користувача (авторизований чи ні), відбувається завдяки зворотному проксі-серверу Ngnix.

4. ТЕСТУВАННЯ СИСТЕМИ І РЕЗУЛЬТАТИ

4.1 Перевірка коректності роботи модулів

Система моніторингу.

Так як мобільний додаток створено за допомогою технології PWA, то у користувачів сучасних браузерів повинна бути можливість встановити додаток на пристрій.

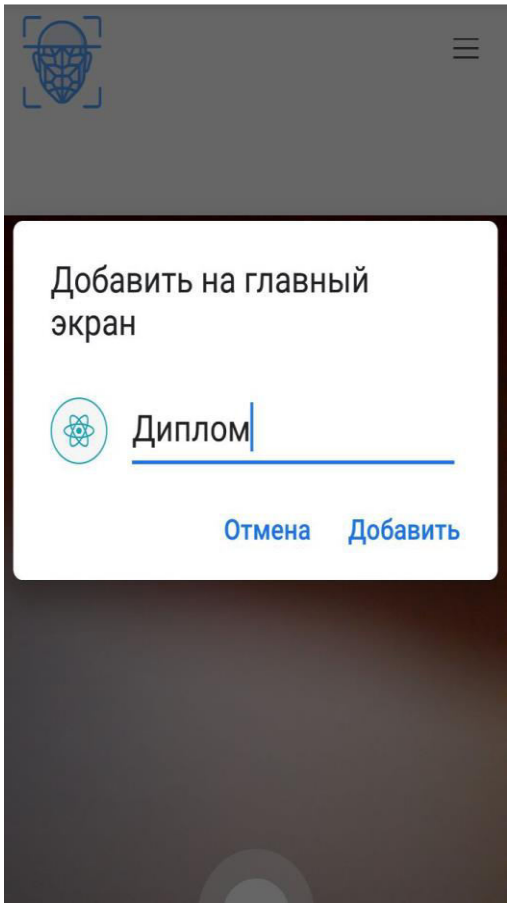

Рисунок 4.1 Вікно для встановлення додатку на пристрій

Після встановлення додатку користувач потрапляє на сторінку з авторизацією. Веб-додаток має адаптивну розмітку, тому в залежності від розмірів екрану комп’ютера або мобільного телефону сторінка авторизації має два різні інтерфейси.

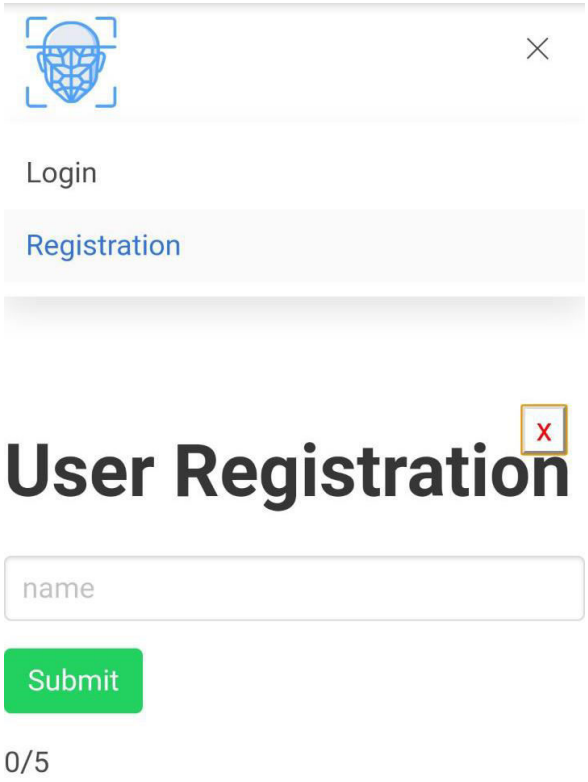

Рисунок 4.2 Стан інтерфейсу, коли їм користуються з мобільного телефону

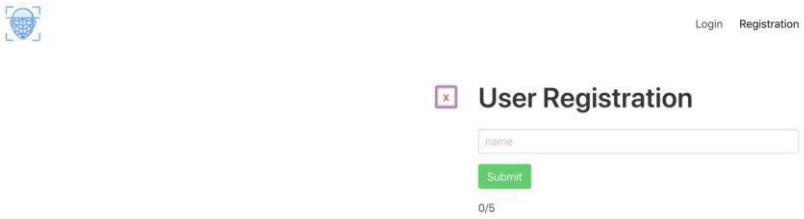

Рисунок 4.3 Стан інтерфейсу, коли їм користуються ноутбука або ПК

Найважливішою частиною функціональності додатку – є коректна робота камери. Незалежно від пристрою (ноутбук або телефон) камера робить знімки і надсилає їх на сервер. На сторінці авторизації і детекторі використовується один компонент, тому повторного тестування на обох сторінках проводити не потрібно. На рисунку 4.5 можна спостерігати форму для реєстрації, в яку вже було додано 5 фотографій і заповнено поле username. Після сабміту форму дані відправляться на сервер і користувача буде додано до бази даних.

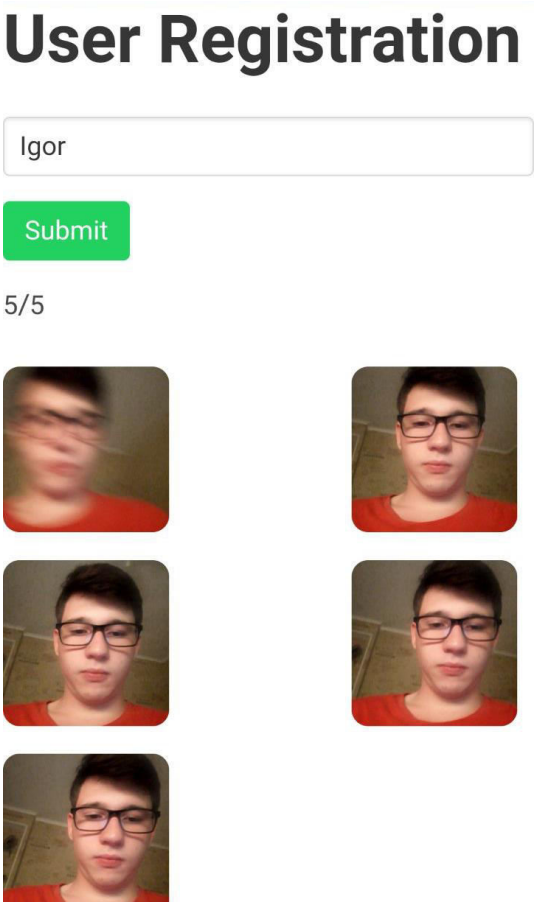

Рисунок 4.5 Форма заповнена фотографіями, які були зроблені за допомогою додатку

На сервер надсилається JSON з ім’ям і фотографіями користувача. Кожне зображення представляє собою base64 строку.


```
{ name: 'igor',
  photos:
   [ ' iVBORw0KGgoAAAANSUhEUgAAAAUA AAAFCAYAAACNbyblAAAAHElEQVQI12P4//8/w
38GIAXDIBKE0DHxgljNBAAO/w38GIAXDIBKE0DHxgljNBAAO/38GIAXDIBKHxgljNBAAO 9TXL0Y4OHwAAAABJRU5Erw38G
IAX38GIAXDIBKHxgljNBAAO 9TXL0Y4OHwAAAABJRU5ErDIBKE0DHxglAAO/w38GIAXDIBKHxgljNBAAO 9TXL0Y4OHwAAA
ABJRU5ErkJggg==',
     '\nvb29t/f3//Ub//ge8WSLf/rhf/3k
dbW1mxsbP//mf///yH5BAAAAAAALAAAAAAQAA4AAA\nRe8L1Ekyky67QZ1hLnjM5UUdeOECwLJoExKcppVOaCcGCmTIHEIU
EqjgaORCMxIC6eOCc\nguWw6aFjsVMkkIr7g77ZKPJjPZqIyd7sJAgVGoEGv2xsBxqNgYPj/gAwXEQA7',
     '" GODlhEAAOALMAAOazToeHh0tLS/7LZv/0j\n    vb29t/f3//Ub//ge8WSLf/rhf
/3kdbW1mxsbP//mf///yH5BAAAAAAALAAAAAAQAA4AAARe8L1\n    Ekyky67QZ1hLnjM5UUdeOECwLJoExKcppVOaCcGC
mT/38GIAXDIBKHxgljNBAAO 9TXL0Y4OHwAAAABJRU5Erw38GIAX38GIAXDIBKHxgljNBAAO 9TXL0Y4OHwAAAABJRU5ErD
IBKE0DHxglAAO/w38GIAXDIBKHxgljNBAAO 9TXL0Y4OHwAAAABJRU5ErkJggg==',
     ' V2VsY29tZSBObyA8Yj5iYXNlNjQuZ3VydTwvAAO 9TXL0Y4OHwAAAABJRU5Erw38GI
AX38GIAXDIBKHxgljNBAAO 9TXL0Y4OHwAAAABJRU5ErDIBKE0DHxglAA' ] }
```


Рисунок 4.6 Приклад JSON об’єкту, який отримує сервер під час реєстрації користувачів

Також, в процесі реєстрації нового користувача створюється модель, за якою система може ідентифікувати власника. Після кожної реєстрації нового

користувача модель оновлюється і записується до бази даних. Якщо в кадрі людина чия модель є в базі даних, то веб-сервіс його ідентифікує. В іншому випадку система оповіщення отримає повідомлення про фіксування непізнаною людини.

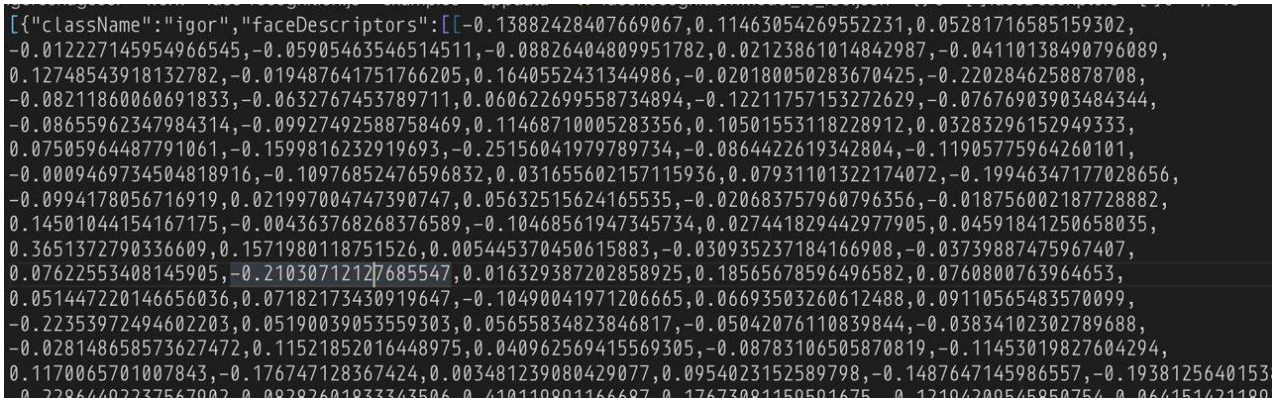

Рисунок 4.6 Модель для ідентифікації користувачів

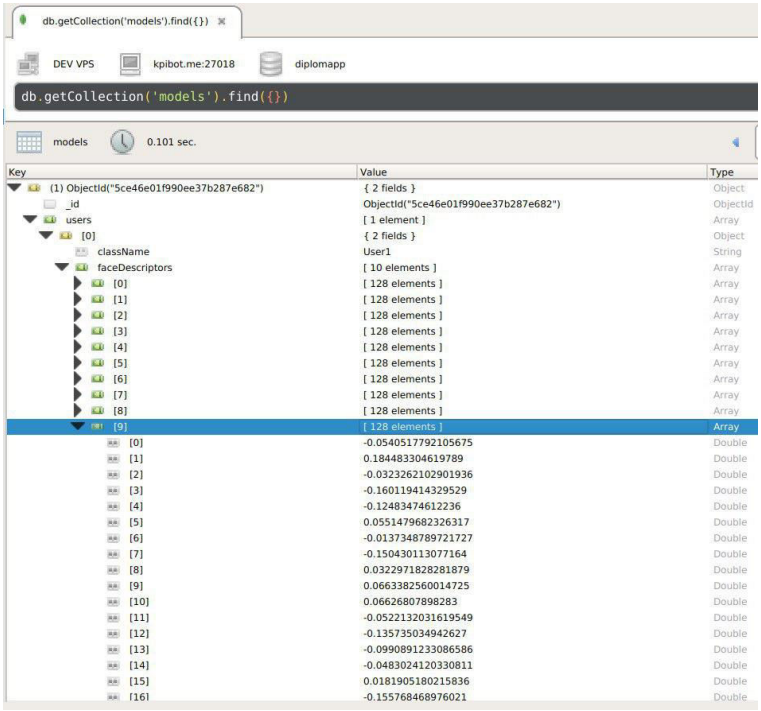

Рисунок 4.7 Колекція з моделями користувачів в базі даних

При отримання повідомлення про виняткову ситуація система оповіщення надсилає до інтерактивної карти, на якій зображується сенсори, які зафіксували порушення.

```
WS:message >>> nobody found!
WS:message >>> nobody found!
WS:message >>> nobody found!
WS:message >>> nobody found!
WS:message >>> nobody found!
WS:message >>> nobody found!
WS:message >>> nobody found!
WS:message >>> nobody found!
WS:message >>> nobody found!
WS:message >>> Detector [50.45, 30.36]
```

Рисунок 4.8 Приклад повідомлень, яка отримує система оповіщення в результаті аналізу зображень

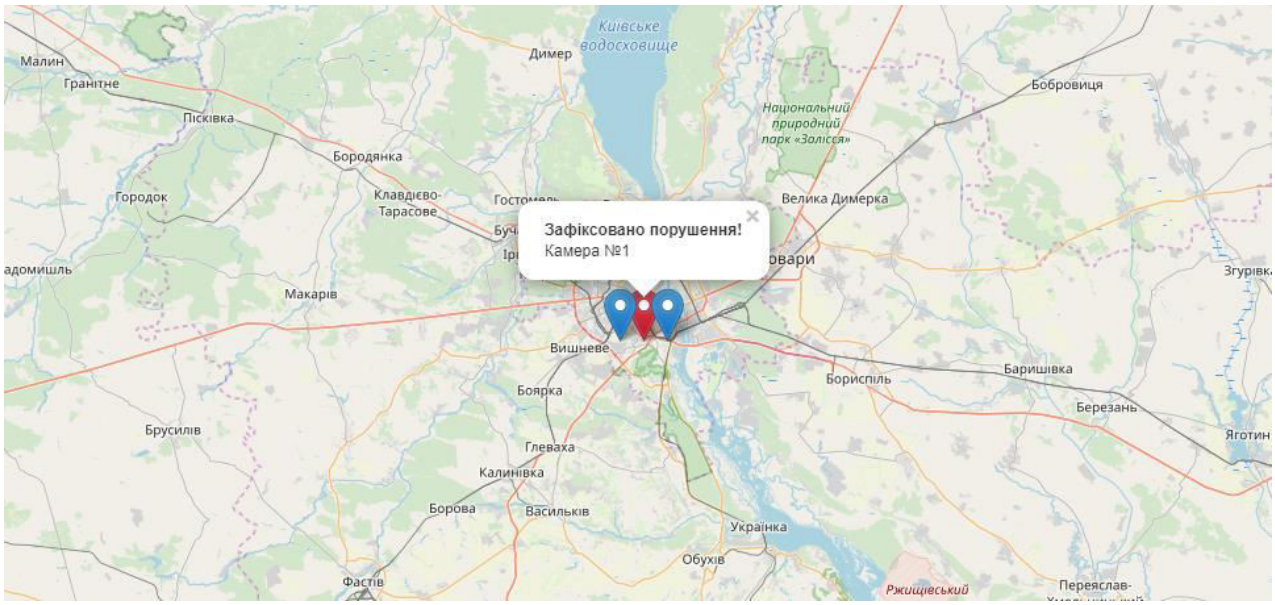

Рисунок 4.8 Приклад повідомлень про пошушення, які відображені на інтерактивній карті

4.2 Швидкодія і гнучкість модулів

Використання підходу SPA(single page application) і асинхронних запитів на серверну частину системи, вдалося створити зручний інтерфейс без перезавантаження сторінки браузера. Для того, щоб забезпечити високу швидкодію і зручну розробку, в дипломному проекті використовуються NodeJS і MongoDB. MongoDB – це документо-орієнтована база даних, яка використовує JSON, як основний формат зберігання. Так як, всі складові системи використовують один формат, то відпадає потреба у трансформації даних, що значно впливає на продуктивність системи. Використання MongoDB також дає можливість до кращого масштабування системи. Сама база даних масштабується горизонтально і використовує для цього техніку сегментування даних – розподіл їх на частини по різними вузлам кластера. Адміністратор або будь-який , хто має доступ до бази, вибирає ключ сегментування, який визначає по якому критерію дані будуть рознесені по вузлам. Завдяки такому підходу, кожний вузол кластера можуть приймати запити і таким чином забезпечується балансування навантаження.

Завдяки використанню мікросервісної архітектури і відсутності сильного зв’язку між модулями, вдалося створити досить піддатливу до змін систему. Якщо користувач має специфічні вимоги або потреби і стандартний функціонал не може його задовільнити, то у такому випадку потрібно додавати нові вузли в систему або підміняти існуючі. Якщо користувачу потрібно додати новий мікросервіс в систему, то для додавання йому необхідно розширити функціональність модулів, які будуть працювати з новим сервісом. Нажаль, постійне нарощування кодової бази при появі нових частин систем – це одна з проблем такої архітектури. Для подміни існуючих модулів, додатку для моніторингу, наприклад, то користувач може використовувати будь-який самописний модуль і надсилати запити до системи сповіщення по відомому йому REST-інтерфейсу.

ВИСНОВКИ

При розробці системи було закладено архітектуру та підходи, які дозволяють легко масштабувати і вдосконалювати систему. Кожний модуль системи представляє собою певну сутність, яка виконує певний функціонал і надає зручний програмний інтерфейс для взаємодії з ним. За відсутності нероздільної і монолітної системи, у користувача є можливість вдосконалювати або змінювати методи моніторингу так, щоб це задовольняло його потребам. Також можна виділити напрямки, які можуть покращити роботу системи:

- Покращення питання безпеки даних, які надсилають користувачі до веб-сервісів.
- Доповнення екосистеми новими модулями, які дозволять користувачам використовувати датчики звуку, світла тощо.
- Розширення методів для оповіщення шляхом відправки повідомлення на телефон або на пошту. Додавання такої можливості може значно розширити цільову аудиторію, які могли б використовувати таку систему.

Система має гарну здатність до масштабування і розширення, проте не позбавлена недоліків і аспектів, з якими можуть виникнути проблеми. Основним критерієм для виборну охоронної системи є метод і об’єкт моніторингу. Якщо у користувача існує потреба в захисту своєї квартири або будинку, то краще скористатися більш популярними рішеннями, які представлені на сучасному ринку охоронних систем. Великий спектр вибору сенсорів і датчиків зможуть надійно зберігати спокій господаря. Проте,

існують ситуації при яких звичайні системи не можуть задовільнити всіх потреб користувача. В таких випадках використання систем з менш зв’язним принципом взаємодії і більш піддатливим до змін інтерфейсу має сенс.

Для створення системи було використано технології, які покращують можливості системи до масштабування і забезпечення більш надійної роботи.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Хабр Microservices. Как правильно делать и когда применять? [Електронний ресурс] URL https://habr.com/ru/company/dataart/blog/280083/ (дата звернення 7.06)
2. Хабр CDN [Електронний ресурс] URL https://habr.com/ru/company/webzilla/blog/236511/ (дата звернення 8.06)
3. Redii Офіційна документація Redis [Електронний ресурс] URL https://redis.io/ (дата звернення 7.06)
4. Establishing a Websocket PUBSUB server with Redis and Asyncio [Електронний ресурс] URL https://yeti.co/blog/establishing-a-websocket-pubsub-server-with-redis-and-asyncio-for-the-light-sensor/ (дата звернення 7.06)
5. Medium How to Scale WebSocket [Електронний ресурс] URL https://hackernoon.com/scaling-websockets-9a31497af051 (дата звернення 8.06)
6. Medium RESTful-api [Електронний ресурс] URL https://medium.com/@lazlojuly/what-is-a-restful-api-fabb8dc2afeb (дата звернення 8.06)
7. Распознавание лиц на основе применения метода виолы–джонса, вейвлет преобразования и метода главных компонент Буй Тхи Тху Чанг, Фан Нгок Хоанг, В.Г. Спицын Томский политехнический университет

8. Хабр Метод Виолы-Джонса (Viola-Jones) как основа для распознавания лиц [Електронний ресурс] URL https://habr.com/ru/post/133826/ (дата звернення 9.06)
9. Medium Карта для React приложения на Leaflet [Електронний ресурс] URL https://medium.com/@eugenebelkovich/%D0%BA%D0%B0%D1%80%D1%82%D0%B0-%D0%B4%D0%BB%D1%8F-react-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F-%D0%BD%D0%B0-leaflet-e6339dc270a4 (дата звернення 10.06)